

- (1) 数据域，用于存放该顶点的信息；
- (2) 指针域，用于指向第一条关联边(弧)的结点。

每一个由边或者弧构成的链表结点，一般需要定义 3 个域：

- (1) 结点域，用于存储该边的另一个端点或者该弧的弧头信息；
- (2) 权值域，用于存储该边(弧)上的权或者其他信息，如果是无权图，则可以不定该域；
- (3) 指针域，用于存储该结点在同一链表中的后继结点的位置。

邻接表存储的空间复杂度为 $O(n+m)$ 。在边或者弧稀疏的情况下，用邻接表存储图比用邻接矩阵存储图要节省存储空间。

代码示例

使用 vector 模拟链表存储一个最多 MAXN 个顶点的无向无权图，代码如下。

```
const int MAXN = 1010;
vector<int> G[MAXN];
void addEdge(int u,int v)
{ // 加入一条边
    G[u].push_back(v);
    G[v].push_back(u); //若无向图,一般需要加双向边
}
void visit(int u)
{ //遍历 u 点所连接的每一个点
    for (int i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i];
    }
}
```

利用结构体存储一个最多 MAXN 个结点的有向有权图，代码如下。

```
const int MAXN = 1010;
struct EDGE
{
    int v,w;
};
vector<EDGE> G[MAXN];
void addEdge(int u,int v,int w)
{ //添加一条有向边 u->v 权值为 w
    EDGE tmp;
    tmp.v = v;
    tmp.w = w;
```

```

        G[u].push_back(tmp);
    }
    void visit(int u)
    { //遍历 u 点所连接的每一个点
        for (int i = 0; i < G[u].size(); i++)
        {
            int v = G[u][i].v;
            int w = G[u][i].w;
        }
    }
}

```

使用链式前向星存储方法，构造最多包含 MAXN 个顶点、MAXM 条边的有向有权图，代码如下。

```

const int MAXN = 1e6 + 5;
const int MAXM = 1e6 + 5;
struct Node
{
    int v, w;
    int next;
} e[MAXM << 1];
int head[MAXN], cnt;
void addEdge(int u, int v, int w)
{
    cnt++;
    e[cnt].v = v;
    e[cnt].w = w;
    e[cnt].next = head[u];
    head[u] = cnt;
}
void visit(int u)
{ //遍历 u 结点链接的所有边
    for (int i = head[u]; i; i = e[i].next)
    {
        int v = e[i].v; //获得 u->v 边的另一端点
        int weight = e[i].w; //获得边的权值
    }
}

```

📖 参考词条

1. 链表：单链表、双向链表、循环链表
2. 栈、队列、链表、向量等容器

延伸阅读

- [1] THOMAS H C, CHARLES E L, RONALD L R, et al. 算法导论(原书第3版)
[M]. 殷建平, 徐云, 王刚, 等译. 北京: 机械工业出版社, 2013: 341-342.
- [2] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 2007: 163-166.
- (佟松龄 叶金毅)

1.4 算 法


1.4.1 算法概念与描述

1.4.1.1 算法概念

在数学和计算机科学之中, 算法(algorithm)是指一个被定义好的、计算机可实施的有限步骤或次序, 常用于计算、数据处理和自动推理。算法是有效方法, 包含一系列定义清晰的指令, 并可于有限的时间及空间内清楚地表述出来。

通常用空间复杂度和时间复杂度来衡量一个算法的优劣, 而同一个问题可以用具有不同时间复杂度和空间复杂度的算法来求解。

算法具有以下五个重要的特征。

- (1) 输入项: 一个算法有 0 个或多个输入。
- (2) 输出项: 一个算法有 1 个或多个输出。
-  (3) 确定性: 算法的每一个步骤必须有确定的定义, 即语句的描述不能存在二义性, 对于每一种情况, 需要执行的动作都应该是严格的、清晰的。
- (4) 有穷性: 算法必须能在执行有限步之后终止, 即在执行完有限步之后自动结束, 而不会出现无限循环, 并且每一步都在规定的时间内完成。
- (5) 可行性: 算法中执行的任何计算步骤都是可以被分解为基本的、可执行的操作步骤, 即算法的每一条运算原则上都能精确地执行。

算法可采用多种描述语言来描述, 各种描述语言在对问题的描述能力方面存在一定的差异。常用的方式有自然语言、流程图、伪代码等。不管用哪种描述方式表达算法, 其描述的结果必须满足算法的五个特征。

延伸阅读

THOMAS H C, CHARLES E L, RONALD L R, et al. 算法导论(原书第3版)[M].

1.4.1.2 算法描述：自然语言描述、流程图描述、伪代码描述

描述算法有很多种方式，其中最常用的描述方式包括自然语言描述、流程图描述和伪代码描述。这些描述方式各有其优缺点，需要根据具体场景选择最合适的描述方式。

1. 自然语言描述

算法的自然语言描述，指的是用日常使用的语言来描述解决问题的具体步骤。

用自然语言描述算法的优点是通俗易懂，当算法中的操作步骤都是顺序执行时，比较直观，容易理解。缺点是如果算法中包含了判断结构和循环结构，并且操作步骤较多时，容易引起歧义，且描述得不够清晰。

2. 流程图描述

流程图是以特定的图形符号加上说明来表示算法的图。流程图用一些图框来表示各种类型的操作，在框内写出各个步骤，然后用带箭头的线将它们连接起来，以表示算法执行的先后顺序。

使用流程图来描述算法，其优点是可以使读者更加清楚地了解整个算法的完整操作过程，有助于在工作过程中及时对算法进行修改。但流程图有其约定的符号，绘制时需要根据其符号进行搭建，绘制过程比较繁琐。

3. 伪代码描述

伪代码是一种非正式的用于描述模块结构图的语言。使用伪代码的目的是使被描述的算法可以容易地以任何一种编程语言实现。因此，伪代码必须结构清晰、代码简单、可读性好。伪代码介于自然语言与编程语言之间，以编程语言的书写形式说明算法功能。使用伪代码，不用拘泥于具体实现，相比程序语言它更类似自然语言，可以将整个算法运行过程的结构用接近自然语言的形式描述出来。

使用伪代码描述算法没有严格的语法限制，书写格式比较自由，只要把意思表达清楚就可以了，它更侧重于对算法本身的描述。在伪代码描述中，表述关键词的语句一般用英文单词，其他语句可以用英文语句，也可以用中文语句。

🔗 参考词条

算法概念

(李绍鸿 谢秋锋)

1.4.2 入门算法

1.4.2.1 枚举法

枚举法(enumeration method)，又称穷举法，指在一个有穷的、可能的解的集合中，枚举出集合中的每一个元素，用题目给定的检验条件来判断该元素是否符合条件，若满

足条件，则该元素为问题的一个解；否则，该元素不是该问题的解。枚举法也是一种搜索算法，即对问题的所有可能解的状态集合进行一次扫描或遍历。

枚举法常用于解决“是否存在”或“有多少种可能”等类型的问题。如，寻找给定范围 $[1, n]$ 内质数个数的問題，可以通过枚举所求范围内的所有自然数，依次判断其是否为质数而求得最终解。

代码示例

求 $[1, n]$ 内质数的个数，主要代码如下。

```
for (int i = 1; i <= n; i++) //枚举 $[1, n]$ 范围内所有的自然数
    if (isprime(i))          //利用 isprime() 函数,判断 i 是否为质数,如果 i 是质数,
                             //则函数返回 1,否则返回 0
        cnt++; //如果 i 是质数,数量加 1
```

参考词条

1. 算法描述：自然语言描述、流程图描述、伪代码描述
2. 搜索算法

典型题目

1. NOIP2008 提高组 火柴棒等式
2. NOIP2010 普及组 数字统计

(李绍鸿 谢秋锋)

1.4.2.2 模拟法

模拟法(simulation method)是一种基本的算法思想，其主要特征是根据给定的规则编写程序，按照时序、逻辑顺序等，对原始计算过程进行细粒度地展现。模拟法一般不会针对计算目标优化或精简计算过程，而是力求尽可能地展现原始计算过程的细节。

代码示例

NOIP2015 提高级题目“神奇的幻方”，即可使用模拟法完成求解。根据题意，确定 1 的位置后，从上至下，按照给定的四个规则，模拟填数字构造幻方的过程，直到 N 个数字都填完即可。题目描述及主要代码实现如下。

幻方是一种很神奇的 $N \times N$ 矩阵，它由数字 $1, 2, 3, \dots, N \times N$ 构成，且每行、每列及两条对角线上的数字之和均相同。

当 N 为奇数时，我们可以通过以下方法构建一个幻方，首先将 1 写在第一行中间，之后按如下方式从小到大依次填写每个数 $K(K=2, 3, \dots, N \times N)$ ：

① 若 $(K-1)$ 在第一行但不在最后一列，则将 K 填在最后一行， $(K-1)$ 所在列的右一列；